(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0054059
(43) 공개일자 2022년05월02일

(51) 국제특허분류(Int. Cl.)
H04L 9/30 (2006.01) *H04L 9/06* (2006.01)
H04L 9/08 (2006.01) *H04L 9/32* (2006.01)
(52) CPC특허분류
H04L 9/3093 (2013.01)
H04L 9/0631 (2013.01)
(21) 출원번호 10-2020-0138524
(22) 출원일자 2020년10월23일
심사청구일자 2020년10월23일

(71) 출원인
상명대학교산학협력단
서울특별시 종로구 홍지문2길 20 (홍지동,상명대학교)
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
김종현
서울특별시 구로구 구로동로 28길 42, 101동 403호
박종환
서울특별시 종로구 홍지문 2길 20 상명대학교 자하관 N505호
(뒷면에 계속)
(74) 대리인
김홍석

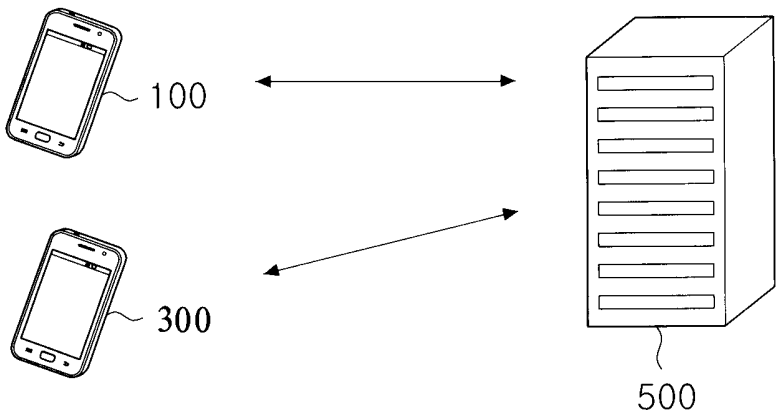
전체 청구항 수 : 총 17 항

(54) 발명의 명칭 **가변 길이 퍼지 데이터를 지원하는 격자 기반 퍼지 추출 장치 및 방법**

(57) 요 약

가변 길이 퍼지 데이터로부터 사용자의 키를 복구하기 위한 헬퍼 데이터를 생성하는 장치 및 방법과 상기 헬퍼 데이터를 이용하여 사용자의 키를 복구하는 장치 및 방법이 개시된다. 상기 헬퍼 데이터를 생성하는 방법은 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되고, 상기 사용자의 제1 퍼지 데이터들을 수신하는 단계, 랜덤 비트열을 선택하는 단계, 모든 인덱스에 대하여, 랜덤 행렬을 선택하는 단계, 에러를 선택하는 단계, 제1 랜덤 비트열을 선택하고 상기 제1 랜덤 비트열을 입력으로 하는 오류 정정 코드(Error Correcting Code, ECC)의 인코딩 알고리즘을 수행하는 단계, 및 상기 랜덤 행렬을 포함하는 제1 헬퍼 데이터를 생성하는 단계를 수행하는 단계, 및 모든 인덱스에 대한 상기 제1 헬퍼 데이터를 포함하는 헬퍼 데이터를 생성하는 단계를 포함한다.

대 표 도 - 도3



10

(52) CPC특허분류
H04L 9/0662 (2013.01)
H04L 9/0894 (2013.01)
H04L 9/3242 (2013.01)
H04L 2209/34 (2013.01)

(72) 발명자
이동훈
서울특별시 종로구 사직로8길 34, 1404호(내수동, 경희궁의아침3단지아파트)
이영경
서울특별시 노원구 노원로1나길 15, 501호
김효승
서울특별시 성북구 안암로9가길 56, 206호

이 발명을 지원한 국가연구개발사업

과제고유번호	1711116146
과제번호	2016-6-00600-005
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보보호핵심원천기술개발(R&D)
연구과제명	(함수암호 1세부) 함수암호 기법 설계·분석 및 구현기술 연구
기 여 율	1/1
과제수행기관명	상명대학교산학협력단
연구기간	2020.02.01 ~ 2021.01.31

명 세 서

청구범위

청구항 1

적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는, 사용자의 퍼지 데이터로부터 키를 복구하는 과정에서 이용되는 헬퍼 데이터를 생성하는 방법에 있어서,

상기 사용자의 제1 퍼지 데이터들($w_1 \in \{0,1\}^{n_1},\cdots,w_i \in 0,1^{n_i},\cdots,w_l \in \{0,1\}^{n_l}$)을 수신하는 단계;

랜덤 비트열($k \in \{0,1\}^{\lambda}$)을 선택하는 단계;

모든 i($i=1,\cdots,l$)에 대하여,

랜덤 행렬($A_i \leftarrow [-L_i, L_i]^{m \times n_i}$)을 선택하는 단계;

에러를 선택하는 단계;

제1 랜덤 비트열($k_i \in \{0,1\}^{\lambda}$)을 선택하고 상기 제1 랜덤 비트열을 입력으로 하는 오류 정정 코드(Error Correcting Code, ECC)의 인코딩 알고리즘을 수행하는 단계; 및

상기 랜덤 행렬을 포함하는 제1 헬퍼 데이터를 생성하는 단계를 수행하는 단계; 및

모든 i에 대한 상기 제1 헬퍼 데이터를 포함하는 헬퍼 데이터를 생성하는 단계를 포함하고,

상기 λ는 보안 상수이고,

상기 l, 상기 L_i, 상기 m, 및 상기 n_i는 임의의 자연수인,

헬퍼 데이터를 생성하는 방법.

청구항 2

제1항에 있어서,

상기 에러를 선택하는 단계는 가우시안 분포(χ)에 대하여 χ_i^m 에서 랜덤하게 벡터 e_i를 선택하는($e_i \leftarrow \chi_i^m$),

헬퍼 데이터를 생성하는 방법.

청구항 3

제2항에 있어서,

상기 제1 헬퍼 데이터($P_i = (A_i, b_i, m_i, c_i)$)를 생성하는 단계는,

제1 수학식을 연산하는 단계;

의사 난수 생성기(PRG)를 이용하여 상기 제1 랜덤 비트열로부터 난수($(k_{i1}, k_{i2}) = \mathbf{PRG}(k_i)$)를 생성하는 단계;

대칭키 암호의 암호화 알고리즘($c_i = \mathbf{SYM.Encrypt}(k_{i1}, k)$)을 수행하는 단계; 및

메시지 인증 코드($m_i = \mathbf{MAC}(k_{i2}, b_i \| c_i)$)를 생성하는 단계를 포함하고,

상기 제1 수학식은 $b_i = A_i w_i + e_i + EC_{d,q}(r_i) \mod q$ 이고,

상기 $EC_{d,q}$는 엠블럼 인코딩(EMBLEM encoding) 기법의 인코딩 함수이고,

에러의 크기를 의미하는 상기 d와 모듈러스를 의미하는 상기 q는 시스템 파라미터로 미리 주어진,

헬퍼 데이터를 생성하는 방법.

청구항 4

제3항에 있어서,

상기 L_i는 q보다 작은 자연수이고,

상기 제1 퍼지 데이터들은 상기 사용자의 지문 데이터 또는 상기 사용자의 홍채 데이터인,

헬퍼 데이터를 생성하는 방법.

청구항 5

제3항에 있어서,

상기 헬퍼 데이터는 $P=(\{A_i,\ b_i,\ m_i,\ c_i\}_{i\in[l]})$인,

헬퍼 데이터를 생성하는 방법.

청구항 6

적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는, 키를 복구하는 방법에 있어서,

사용자의 제2 퍼지 데이터($w' \in \{0,1\}^n$)와 헬퍼 데이터($P=(\{A_i,\ b_i,\ m_i,\ c_i\}_{i\in[l]})$)를 수신하는 단계;

상기 헬퍼 데이터에서 $n_i = n$을 만족하는 첫 번째 인덱스 i를 선택하는 단계;

제1 수학식을 연산하는 단계;

상기 제1 수학식의 연산 결과를 입력으로 하는 오류 정정 코드의 디코딩 알고리즘을 수행하여 제2 비트열($k'_i = \mathbf{ECC.Decode}(r'_i)$)을 생성하는 단계;

의사 난수 생성기를 이용하여 상기 제2 비트열로부터 난수($(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i)$)를 생성하는 단계;

메시지 인증 코드($m_i' = \mathbf{MAC}(k_{i2}', b_i \| c_i)$)를 생성하는 단계; 및

상기 메시지 인증 코드와 상기 헬퍼 데이터를 비교한 후 대칭키 암호의 복호화 알고리즘을 이용하여 제1 비트열($k' = \mathbf{SYM.Decrypt}(k'_{i1}, c_i)$)을 생성하는 단계를 포함하고,

상기 제1 수학식은 $r'_i = DC_q(b_i - A_i w' \bmod q)$이고, 상기 DC_q는 엠블럼 인코딩 기법의 디코딩 함수이고,

모듈러스를 의미하는 상기 q는 임의의 자연수인,

키를 복구하는 방법.

청구항 7

제6항에 있어서,

상기 첫 번째 인덱스 i를 선택하는 단계에서 i가 존재하지 않는 경우, 상기 사용자의 제3 퍼지 데이터를 수신하고 상기 헬퍼 데이터에서 $n_i = n$을 만족하는 첫 번째 인덱스 i를 선택하는,

키를 복구하는 방법.

청구항 8

제6항에 있어서,

상기 제1 비트열을 생성하는 단계는, 상기 메시지 인증 코드(m_i')와 상기 헬퍼 데이터에 포함된 m_i가 동일한 경우에만 상기 제1 비트열을 생성하는,

키를 복구하는 방법.

청구항 9

제6항에 있어서,

상기 헬퍼 데이터를 생성하는 과정 중에 사용된 상기 사용자의 제1 퍼지 데이터들 중 i번째 퍼지 데이터와 상기 제2 퍼지 데이터의 거리가 임계치보다 작은 경우, 상기 헬퍼 데이터를 생성하는 과정 중에 생성된 랜덤 비트열과 상기 제1 비트열은 동일한 값을 갖는,

키를 복구하는 방법.

청구항 10

적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는, 사용자의 퍼지 데이터로부터 키를 복구하는 과정에서 이용되는 헬퍼 데이터를 생성하는 방법에 있어서,

상기 사용자의 제1 퍼지 데이터들($w_1 \in \{0,1\}^{n_1},\cdots,w_i \in 0,1^{n_i},\cdots,w_l \in \{0,1\}^{n_l}$)을 수신하는 단계;

랜덤 비트열($k \in \{0,1\}^{\lambda}$)을 선택하는 단계;

모든 i($i=1,\cdots,l$)에 대하여,

랜덤 행렬($A_i \leftarrow [-L_i, L_i]^{m \times n_i}$)을 선택하는 단계;

제1 랜덤 비트열($k_i \in \{0,1\}^{\lambda}$)을 선택하고 상기 제1 랜덤 비트열을 입력으로 하는 오류 정정 코드(Error Correcting Code, ECC)의 인코딩 알고리즘($r_i = \mathbf{ECC.Encode}(k_i)$)을 수행하는 단계; 및

상기 랜덤 행렬을 포함하는 제1 헬퍼 데이터를 생성하는 단계를 수행하는 단계; 및

모든 i에 대한 상기 제1 헬퍼 데이터를 포함하는 헬퍼 데이터를 생성하는 단계를 포함하고,

상기 λ는 보안 상수이고,

상기 l, 상기 L_i, 상기 m, 및 상기 n_i는 임의의 자연수인,

헬퍼 데이터를 생성하는 방법.

청구항 11

제10항에 있어서,

상기 제1 헬퍼 데이터($P_i = (A_i, b_i, m_i, c_i)$)를 생성하는 단계는,

제1 수학식을 연산하는 단계;

의사 난수 생성기(PRG)를 이용하여 상기 제1 랜덤 비트열로부터 난수($(k_{i1}, k_{i2}) = \mathbf{PRG}(k_i)$)를 생성하는 단계;

대칭키 암호의 암호화 알고리즘($c_i = \mathbf{SYM.Encrypt}(k_{i1}, k)$)을 수행하는 단계; 및

메시지 인증 코드($m_i = \mathbf{MAC}(k_{i2}, b_i \| c_i)$)를 생성하는 단계를 포함하고,

상기 제1 수학식은 $b_i = \lfloor A_i w_i \bmod q \rfloor_p + EC_{d,p}(r_i) \bmod p$ 이고,

상기 $EC_{d,p}$는 엠블럼 인코딩(EMBLEM encoding) 기법의 인코딩 함수이고,

에러의 크기를 의미하는 상기 d와 모듈러스를 의미하는 상기 p는 시스템 파라미터로 미리 주어진,

헬퍼 데이터를 생성하는 방법.

청구항 12

제11항에 있어서,

상기 L_i는 q 보다 작은 자연수이고,

상기 제1 퍼지 데이터들은 상기 사용자의 지문 데이터 또는 상기 사용자의 홍채 데이터인,

헬퍼 데이터를 생성하는 방법.

청구항 13

제11항에 있어서,

상기 헬퍼 데이터는 $P = (\{A_i,\ b_i,\ m_i,\ c_i\}_{i \in [l]})$ 인,

헬퍼 데이터를 생성하는 방법.

청구항 14

적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는, 키를 복구하는 방법에 있어서,

사용자의 제2 퍼지 데이터($w' \in \{0,1\}^n$)와 헬퍼 데이터($P = (\{A_i,\ b_i,\ m_i,\ c_i\}_{i \in [l]})$)를 수신하는 단계;

상기 헬퍼 데이터에서 $n_i = n$을 만족하는 첫 번째 인덱스 i를 선택하는 단계;

제1 수학식을 연산하는 단계;

상기 제1 수학식의 연산 결과를 입력으로 하는 오류 정정 코드의 디코딩 알고리즘을 수행하여 제2 비트열($k'_i = \mathbf{ECC.Decode}(r'_i)$)을 생성하는 단계;

의사 난수 생성기를 이용하여 상기 제2 비트열로부터 난수($(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i)$)를 생성하는 단계;

메시지 인증 코드($m_i' = \mathbf{MAC}(k_{i2}', b_i \| c_i)$)를 생성하는 단계; 및

상기 메시지 인증 코드와 상기 헬퍼 데이터를 비교한 후 대칭키 암호의 복호화 알고리즘을 이용하여 제1 비트열($k' = \mathbf{SYM.Decrypt}(k'_{i1}, c_i)$)을 생성하는 단계를 포함하고,

상기 제1 수학식은 $r'_i = DC_q\left(\left(\frac{q}{p}\right) b_i - A_i w' \bmod q\right)$ 이고, 상기 DC_q는 엠블럼 인코딩 기법의 디코딩 함수이고,

모듈러스를 의미하는 상기 q는 임의의 자연수인,

키를 복구하는 방법.

청구항 15

제14항에 있어서,

상기 첫 번째 인덱스 i를 선택하는 단계에서 i가 존재하지 않는 경우, 상기 사용자의 제3 퍼지 데이터를 수신하고 상기 헬퍼 데이터에서 $n_i = n$을 만족하는 첫 번째 인덱스 i를 선택하는,

키를 복구하는 방법.

청구항 16

제14항에 있어서,

상기 제1 비트열을 생성하는 단계는, 상기 메시지 인증 코드(m_i')와 상기 헬퍼 데이터에 포함된 m_i가 동일한 경우에만 상기 제1 비트열을 생성하는,

키를 복구하는 방법.

청구항 17

제14항에 있어서,

상기 헬퍼 데이터를 생성하는 과정 중에 사용된 상기 사용자의 제1 퍼지 데이터들 중 i번째 퍼지 데이터와 상기 제2 퍼지 데이터의 거리가 임계치보다 작은 경우, 상기 헬퍼 데이터를 생성하는 과정 중에 생성된 랜덤 비트열과 상기 제1 비트열은 동일한 값을 갖는,

키를 복구하는 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 퍼지 추출기(Fuzzy Extractor)에 관한 것으로, 특히 격자(Lattice)를 이용하여 추가적인 비밀 정보 없이 노이즈(Noise)가 포함된 가변 길이 퍼지 데이터로부터 암호학적인 키로 사용 가능한 동일한 랜덤 비트열(Bit string)을 안전하고 효율적으로 추출하는 장치 및 방법에 관한 것이다.

배 경 기 술

[0002] **LWE 문제(Learning with Errors problem).**

[0003] LWE 문제는 2005년 Regev에 의해 처음 소개되었다. m, n, k, q는 양의 정수이고 D_s와 D_e는 $\mathbb{Z}_q$ 상의 임의의 분포라고 하자. 랜덤하게 선택된 행렬 $A \in \mathbb{Z}_q^{m \times n}$, $S \in D_s^{n \times k}$, 및 $E \in D_e^{m \times k}$에 대하여, $B = AS + E \,(mod\, q)$를 계산하자. (A, B)가 주어졌을 때, S를 찾는 것을 LWE 문제라고 정의한다. 일반적으로 D_s는 $\mathbb{Z}_q$를, D_e는 표준편차가 s인 가우시안 분포 GD_s를 의미한다.

[0004] **LWR 문제(Learning with Rounding problem).**

[0005] LWR 문제는 2012년 Banerjee, Peikert, 그리고 Rosen에 의해 처음 소개되었다. LWE 문제의 변형으로, 차이점은 $AS \in \mathbb{Z}_q^{m \times k}$의 정확한 값을 숨기기 위해 LWE 문제에서는 가우시안 분포에서 선택한 작은 에러를 더해주는데, 이 부분을 대신하여 라운딩 함수 $\lfloor AS \rceil_p : \mathbb{Z}_q \rightarrow \mathbb{Z}_p \;\; (p < q)$를 연산하는 것이다. 동일한 A와 S에 대해서는 항상 동일한 값이 출력되기 때문에, 결정적인(Deterministic) 에러가 더해지는 것으로도 볼 수 있다. m, n, k, q, 및 p는 양의 정수이고, D_s는 $\mathbb{Z}_q$ 상의 임의의 분포라고 하자. 랜덤하게 선택된 행렬 $A \in \mathbb{Z}_q^{m \times n}$와 $S \in D_s^{n \times k}$에 대하여, $B = \lfloor AS (\mathrm{mod}\, q) \rceil_p$를 계산하자. (A, B)가 주어졌을 때, S를 찾는 것을 LWR 문제라고 정의한다.

[0006] **퍼지 추출기(Fuzzy Extractor, FE).**

[0007] 퍼지 추출기는 (임계치 이하의) 노이즈가 포함된 두 개의 데이터로부터 동일한 랜덤 비트열을 추출하는 기술이

다. 퍼지 데이터의 대표적인 예로는 사용자의 생체 정보가 있으며, 퍼지 추출기를 통해 출력된 비트열은 암호 기법이나 인증 프로토콜에서 사용자의 비밀키로 사용될 수 있다. 퍼지 추출기 FE = (Gen, Rec)는 두 개의 알고리즘으로 구성된다.

[0008] - **생성(Gen).** 퍼지 데이터 w를 입력으로 받아 헬퍼 데이터(Helper data) P와 랜덤 비트열 $k \in \{0,1\}^{\lambda}$를 출력한다. 여기서, λ는 보안 상수를 의미할 수 있다.

[0009] - **복원(Rec).** 퍼지 데이터 w'를 입력으로 받아 비트열 $k' \in \{0,1\}^{\lambda}$를 출력한다.

[0010] (Correctness) 두 퍼지 데이터 w와 w'의 거리를 $d(w,w')$, 임계치(Threshold value)를 t라고 하자. $d(w,w') \le t$를 만족하는 임의의 두 퍼지 데이터 w, w'와 Gen(w)을 통해 생성된 (P,k)에 대하여, $\mathbf{Rec}(P,w')=k$이 성립한다.

[0011] <u>**오류 정정 코드(Error Correcting Code, ECC).**</u>

[0012] 오류 정정 코드는 데이터를 전송하는 과정에서 오류가 발생했을 때 오류를 찾아내서 원래 값으로 복원할 수 있도록 하는 기법이다. 비트열에서의 오류 정정 코드의 경우, 1비트를 표현하기 위하여 1비트 이상의 중복(Redundant) 혹은 패리티(Parity) 비트를 사용함으로써 전송 과정에서 발생하는 오류를 탐지하고 정정할 수 있도록 한다. 오류 정정 코드 ECC = (Encode, Decode)는 두 개의 알고리즘으로 구성된다.

[0013] - **인코딩(Encode).** 비트열 $k \in \{0,1\}^{\lambda}$을 입력으로 받아 $m(>\lambda)$비트 코드워드(Codeword) $C \in \{0,1\}^{m}$를 출력한다. 여기서, λ는 보안 상수를 의미할 수 있다.

[0014] - **디코딩(Decode).** 코드워드 $C' \in \{0,1\}^{m}$을 입력으로 받아 길이가 λ인 비트열 $k' \in \{0,1\}^{\lambda}$을 출력한다.

[0015] (Correctness) 두 코드워드 C, C'의 해밍 거리(Hamming distance)를 $HD(C,C')$, ECC로 처리 가능한 최소 해밍 거리를 d라고 하자. $\mathbf{Encode}(k)$를 통해 생성된 코드워드 C와 $HD(C,C') \le d$를 만족하는 임의의 코드워드 C'에 대하여, $\mathbf{Decode}(C')=k$이 성립한다.

[0016] <u>**엠블럼 인코딩(EMBLEM encoding).**</u>

[0017] LWE와 LWR 문제를 기반으로 설계되는 기법의 경우 에러(Error)가 공개 정보에 포함되어 있기 때문에, 정확한 결과값을 얻기 위해서는 이를 처리하기 위한 인코딩/디코딩 방법이 추가적으로 필요하다. LWE 또는 LWR 문제를 기반으로 하여 설계되는 퍼지 추출기의 경우 공개된 헬퍼 데이터에 에러가 포함되며, 키를 복원(Rec)하는 과정에서 해당 에러를 제거하고 처음 생성(Gen)한 키를 정확하게 얻기 위하여 인코딩/디코딩이 사용된다. 본 발명에서는 엠블럼 인코딩/디코딩 방법을 사용한다(비특허문헌 1 참조).

[0018] - 본 발명에서 사용하는 엠블럼 인코딩 함수는 길이가 m인 비트열 $r \in \{0,1\}^{m}$, 키를 복원할 때 발생하는 에러의 크기 d, 그리고 모듈러스(Modulus) q를 입력으로 받아 $\mathbb{Z}_q^m$ 상의 벡터 R을 출력한다. 엠블럼 디코딩 함수는 인코딩의 역함수로, 벡터 $R \in \mathbb{Z}_q^m$과 모듈러스 q를 입력으로 받아 벡터 $r \in \{0,1\}^{m}$을 출력한다.

[0019] □ **엠블럼 인코딩(*EC*).** (r,d,q)를 입력으로 받아 $m \times 1$ 벡터 R을 출력한다.

[0020] - 입력값인 비트열 r의 길이를 m이라고 하고, $r=(r_1,\ldots,r_m) \in \{0,1\}^{m}$이라고 하자. 모든 $i \in [1,m]$에 대하여, $R_i = (r_i \| 1 \| 0^d) \in \mathbb{Z}_q$를 계산한다.

[0021] - $m \times 1$ 벡터 $R=(R_1,\ldots,R_m) \in \mathbb{Z}_q^m$을 출력한다.

[0022] □ **엠블럼 디코딩(*DC*).** (R,q)를 입력으로 받아 벡터 r을 출력한다.

[0023] - 입력값인 벡터 R을 $R=(R_1,\ldots,R_m)\in\mathbb{Z}_q^m$이라고 하자. 모든 $i\in[1,m]$에 대하여, R_i의 최상위 1비트를 $r_i\in\{0,1\}$로 설정한다.

[0024] - $m\times 1$ 벡터 $r=(r_1,\ldots,r_m)\in\{0,1\}^m$을 출력한다.

[0025] <u>의사 난수 생성기(Pseudo-random Generator).</u>

[0026] 의사 난수 생성기는 실제 랜덤과 구분하기 어려운 비트열을 생성하는 알고리즘이다. 의사 난수 생성기 PRG는 랜덤 비트열 $r\in\{0,1\}^\lambda$을 입력받아 비트열 $s=\mathbf{PRG}(r)\in\{0,1\}^{\lambda'}$를 출력한다. 이때 $\lambda'>\lambda$를 만족하며, 비트열 s는 $\{0,1\}^{\lambda'}$에서 랜덤하게 선택한 비트열과 다항 시간 안에 구분하기 어렵다.

[0027] <u>메시지 인증 코드(Message Authentication Code, MAC).</u>

[0028] 메시지 인증 코드는 데이터 전송에서 오류가 발생하였는지 확인할 수 있는 알고리즘이다. 메시지 인증 코드 MAC는 일반적인 해시함수와 다르게 함수의 입력으로 비밀키 k가 필요하며, 메시지 m에 대해 비밀키 k를 이용하여 메시지 인증 코드 $c=\mathbf{MAC}(k,m)$을 생성한다. 데이터에 오류가 발생하였는지 확인을 위해서는 비밀키 k에 대해 최초에 생성한 메시지 인증 코드 $c=\mathbf{MAC}(k,m)$와 추후 전달받은 메시지 m'에 대한 메시지 인증 코드 $c'=\mathbf{MAC}(k,m')$를 비교함으로써 확인할 수 있다.

[0029] <u>대칭키 암호(Symmetric-key Encryption).</u>

[0030] 대칭키 암호는 암호 알고리즘의 한 종류로, 암호화와 복호화에 같은 비밀키를 쓰는 암호화 알고리즘이다. 대칭키 암호 $\mathbf{SYM}=\mathbf{(Encrypt,\ Decrypt)}$는 두 개의 알고리즘으로 구성된다.

[0031] - 암호화(Encrypt). 비밀키 $k\in K$와 메시지 $m\in M$을 입력받아 암호문 $c\in M$를 출력한다.

[0032] - 복호화(Decrypt). 비밀키 $k\in K$와 암호문 $c\in M$를 입력받아 메시지 $m\in M$을 출력한다.

[0033] (Correctness) 비밀키 $k\in K$와 메시지 $m\in M$으로 생성한 암호문 $c=\mathbf{Encrypt}(k,m)\in C$에 대하여 $m=\mathbf{Decrypt}(k,\ \mathbf{Encrypt}(k,m))$를 만족한다.

[0034] 본 발명에서는 상술한 기법들을 이용하여 추가적인 비밀 정보 없이 노이즈가 있는 가변 길이 퍼지 데이터로부터 암호학적인 키로 사용이 가능한 동일한 랜덤 비트열을 안전하고 효율적으로 추출하는 방법을 제안하고자 한다.

선행기술문헌

특허문헌

[0035] (특허문헌 0001) 대한민국 공개특허 제2007-0086656호 (2007.08.27. 공개)

(특허문헌 0002) 대한민국 공개특허 제2017-0087755호 (2017.07.31. 공개)

비특허문헌

[0036] (비특허문헌 0001) M. S대 S. Kim, D.H. Lee, and J.H. Park, "EMBLEM: (R)LWE-based key encapsulation with a new multi-bit encoding method", Int. J. Inf. Secur. (2019).

발명의 내용

해결하려는 과제

[0037] 본 발명이 이루고자 하는 기술적인 과제는 생체 정보와 같이 측정할 때마다 노이즈가 발생하는 가변 길이 퍼지 데이터를 이용하여, 추가적인 비밀 정보를 사용하지 않고도 암호학적 키로 사용할 수 있는 동일한 랜덤 비트열을 출력하는 퍼지 추출 장치 및 방법을 제공하는 것이다.

과제의 해결 수단

[0038] 본 발명의 일 실시예에 따른, 사용자의 퍼지 데이터로부터 키를 복구하는 과정에서 이용되는 헬퍼 데이터를 생성하는 방법은 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되고, 상기 사용자의 제1 퍼지 데이터들($w_1 \in \{0,1\}^{n_1},\cdots,w_i \in 0,1^{n_i},\cdots,w_l \in \{0,1\}^{n_l}$)을 수신하는 단계, 랜덤 비트열($k \in \{0,1\}^{\lambda}$)을 선택하는 단계, 모든 i($i=1,\cdots,l$)에 대하여, 랜덤 행렬($A_i \leftarrow [-L_i, L_i]^{m \times n_i}$)을 선택하는 단계, 에러를 선택하는 단계, 제1 랜덤 비트열($k_i \in \{0,1\}^{\lambda}$)을 선택하고 상기 제1 랜덤 비트열을 입력으로 하는 오류 정정 코드(Error Correcting Code, ECC)의 인코딩 알고리즘을 수행하는 단계, 및 상기 랜덤 행렬을 포함하는 제1 헬퍼 데이터를 생성하는 단계를 수행하는 단계, 및 모든 i에 대한 상기 제1 헬퍼 데이터를 포함하는 헬퍼 데이터를 생성하는 단계를 포함하고, 상기 λ는 보안 상수이고, 상기 l, 상기 L_i, 상기 m, 및 상기 n_i는 임의의 자연수이다.

[0039] 본 발명의 일 실시예에 따른, 키를 복구하는 방법은 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되고, 사용자의 제2 퍼지 데이터($w' \in \{0,1\}^{n}$)와 헬퍼 데이터($P=(\{A_i,\ b_i,\ m_i,\ c_i\}_{i\in[l]})$)를 수신하는 단계, 상기 헬퍼 데이터에서 $n_i = n$을 만족하는 첫 번째 인덱스 i를 선택하는 단계, 제1 수학식을 연산하는 단계, 상기 제1 수학식의 연산 결과를 입력으로 하는 오류 정정 코드의 디코딩 알고리즘을 수행하여 제2 비트열($k'_i = \mathbf{ECC.Decode}(r'_i)$)을 생성하는 단계, 의사 난수 생성기를 이용하여 상기 제2 비트열로부터 난수($(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i)$)를 생성하는 단계, 메시지 인증 코드($m_i' = \mathbf{MAC}(k_{i2}', b_i \| c_i)$)를 생성하는 단계, 및 상기 메시지 인증 코드와 상기 헬퍼 데이터를 비교한 후 대칭키 암호의 복호화 알고리즘을 이용하여 제1 비트열($k' = \mathbf{SYM.Decrypt}(k'_{i1}, c_i)$)을 생성하는 단계를 포함하고, 상기 제1 수학식은 $r'_i = DC_q(b_i - A_i w' \bmod q)$이고, 상기 DC_q는 엠블럼 인코딩 기법의 디코딩 함수이고, 모듈러스를 의미하는 상기 q는 임의의 자연수이다.

[0040] 본 발명의 다른 실시예에 따른, 사용자의 퍼지 데이터로부터 키를 복구하는 과정에서 이용되는 헬퍼 데이터를 생성하는 방법은 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되고, 상기 사용자의 제1 퍼지 데이터들($w_1 \in \{0,1\}^{n_1},\cdots,w_i \in 0,1^{n_i},\cdots,w_l \in \{0,1\}^{n_l}$)을 수신하는 단계, 랜덤 비트열($k \in \{0,1\}^{\lambda}$)을 선택하는 단계, 모든 i($i=1,\cdots,l$)에 대하여, 랜덤 행렬($A_i \leftarrow [-L_i, L_i]^{m \times n_i}$)을 선택하는 단계, 제1 랜덤 비트열($k_i \in \{0,1\}^{\lambda}$)을 선택하고 상기 제1 랜덤 비트열을 입력으로 하는 오류 정정 코드(Error Correcting Code, ECC)의 인코딩 알고리즘($r_i = \mathbf{ECC.Encode}(k_i)$)을 수행하는 단계, 및 상기 랜덤 행렬을 포함하는 제1 헬퍼 데이터를 생성하는 단계를 수행하는 단계, 및 모든 i에 대한 상기 제1 헬퍼 데이터를 포함하는 헬퍼 데이터를 생성하는 단계를 포함하고, 상기 λ는 보안 상수이고, 상기 l, 상기 L_i, 상기 m, 및 상기 n_i는 임의의 자연수이다.

[0041] 본 발명의 다른 실시예에 따른, 키를 복구하는 방법은 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되고, 사용자의 제2 퍼지 데이터($w' \in \{0,1\}^{n}$)와 헬퍼 데이터($P=(\{A_i,\ b_i,\ m_i,\ c_i\}_{i\in[l]})$)를 수신하는 단계, 상기 헬퍼 데이터에서 $n_i = n$을 만족하는 첫 번째 인덱스 i를 선택하는 단계, 제1 수학식을 연산하는 단계, 상기 제1 수학식의 연산 결과를 입력으로 하는 오류 정정 코드의 디코딩 알고리즘을 수행하여 제2 비트열($k'_i = \mathbf{ECC.Decode}(r'_i)$)을 생성하는 단계, 의사 난수 생성기를 이용하여 상기 제2 비트열로부터 난수($(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i)$)를 생성하는 단계, 메시지 인증 코드($m_i' = \mathbf{MAC}(k_{i2}', b_i \| c_i)$)를 생성하는 단계, 및 상기 메시지 인증 코드와 상기 헬퍼 데이터를 비교한 후 대칭키 암호의 복호화 알고리즘을 이용하여 제1 비트열

($k' = \mathbf{SYM.Decrypt}(k'_{il}, c_i)$)을 생성하는 단계를 포함하고, 상기 제1 수학식은 $r'_i = DC_q(\left(\frac{q}{p}\right)b_i - A_i w' \bmod q)$ 이고, 상기 DC_q는 엠블럼 인코딩 기법의 디코딩 함수이고, 모듈러스를 의미하는 상기 q는 임의의 자연수이다.

발명의 효과

[0042] 본 발명의 실시예에 따른 퍼지 추출 장치 및 방법에 의할 경우, 측정시마다 노이즈가 발생하는 가변 길이 퍼지 데이터를 이용하여 암호학적 키로 사용이 가능한 동일한 랜덤 비트열을 생성할 수 있다.

[0043] 또한, 본 발명에 의할 경우, 임계치 범위 내에 속하는 가변 길이 퍼지 데이터에 대해서만 동일한 랜덤 비트열을 생성하고 임계치를 초과하는 노이즈가 포함된 데이터에 대해서는 서로 다른 비트열이 출력된다.

[0044] 또한, 본 발명에 의할 경우, 양자 컴퓨팅 환경에서도 공개된 정보와 출력된 비트열로부터 입력값인 퍼지 데이터의 정보가 노출되지 않는 효과가 있다.

도면의 간단한 설명

[0045] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 상세한 설명이 제공된다.

도 1은 본 발명의 일 실시예에 따른 LWE 기반의 퍼지 추출 기법을 설명하기 위한 흐름도이다.

도 2는 본 발명의 다른 실시예에 따른 LWR 기반의 퍼지 추출 기법을 설명하기 위한 흐름도이다.

도 3은 본 발명의 일 실시예에 따른 퍼지 추출 시스템을 도시한다.

도 4는 도 3에 도시된 제1 단말의 기능 블럭도이다.

도 5는 도 3에 도시된 제2 단말의 기능 블럭도이다.

발명을 실시하기 위한 구체적인 내용

[0046] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예들은 다양한 형태들로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.

[0047] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에서 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시예들을 특정한 개시 형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물, 또는 대체물을 포함한다.

[0048] 제1 또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않은 채, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고 유사하게 제2 구성 요소는 제1 구성 요소로도 명명될 수 있다.

[0049] 어떤 구성 요소가 다른 구성 요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성 요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성 요소가 다른 구성 요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는 중간에 다른 구성 요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성 요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0050] 본 명세서에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로서, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 본 명세서에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야

한다.

[0051] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0052] 이하, 본 명세서에 첨부된 도면들을 참조하여 본 발명의 실시예들을 상세히 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0054] 본 발명에서, 퍼지 데이터(Fuzzy data)는 가변 길이의 비트열 형태로 표현된다. 지문 정보, 홍채 정보 등과 같은 퍼지 데이터는 사용자의 고유한 값으로, 사용자의 프라이버시 보호를 위해 퍼지 데이터는 시스템 전체적으로 노출되지 않도록 하여야 한다. 또한, 본 발명에서는 가변 길이의 비트열의 형태를 가진 퍼지 데이터를 처리하기 위해 초기 입력 단계에서 입력 기기를 여러 번 이용하여 다양한 길이의 퍼지 데이터를 입력받을 수 있다.

[0055] 길이가 n(n은 임의의 자연수)인 두 벡터 $w=(w_1,\dots,w_n)\in\{0,1\}^n$와 $w'=(w'_1,\dots,w'_n)\in\{0,1\}^n$의 거리(distance)는 $d(w,w')=\sum_{i=1}^{n}|w_i-w'_i|$로 정의된다. 즉, 벡터 형태로 표현되는 두 개의 퍼지 데이터의 거리는 서로 상이한 값을 갖는 대응되는 비트의 개수를 의미할 수 있다. 임계치 t는 0과 n 사이의 정수이다. 본 발명에서 두 개의 퍼지 데이터의 거리가 임계치보다 작은 경우, 두 개의 퍼지 데이터로부터 동일한 랜덤 비트열이 생성되어야 한다.

[0056] 본 발명에서는 오류 정정 코드 $\mathbf{ECC}=(\mathbf{ECC.Encode},\ \mathbf{ECC.Decode})$를 사용한다. $\mathbf{ECC.Encode}$ 알고리즘(오류 정정 코드의 인코딩 알고리즘)은 길이가 λ인 비트열을 입력으로 받아 길이가 $m(>\lambda)$인 비트열을 출력하고, $\mathbf{ECC.Decode}$ 알고리즘(오류 정정 코드의 디코딩 알고리즘)은 반대로 길이가 m인 비트열을 입력받아 길이가 λ인 비트열을 출력한다.

[0057] 또한, 본 발명에서는 의사 난수 생성기 $\mathbf{PRG}$를 사용한다. $\mathbf{PRG}$는 길이가 λ인 비트열을 입력으로 받아 길이가 2λ인 비트열을 출력한다.

[0058] 또한, 본 발명에서는 메시지 인증 코드 $\mathbf{MAC}$를 사용한다. $\mathbf{MAC}$는 길이가 λ인 비트열을 비밀키로 사용하여 데이터에 대한 메시지 인증 코드를 생성한다.

[0059] 또한, 본 발명에서는 대칭키 암호 $\mathbf{SYM}=(\mathbf{SYM.Encrypt},\ \mathbf{SYM.Decrypt})$를 사용한다. $\mathbf{SYM.Encrypt}$ 알고리즘(대칭키 암호의 암호화 알고리즘)과 $\mathbf{SYM.Decrypt}$ 알고리즘(대칭키 암호의 복호화 알고리즘)은 길이가 λ인 비트열을 비밀키로 사용하여 메시지에 대한 암호문을 생성하거나, 암호문을 복호화하여 메시지를 복원한다.

[0060] 본 발명에서는 (1) LWE 문제를 기반으로 설계된 퍼지 추출기와 (2) LWR 문제를 기반으로 설계된 퍼지 추출기를 제안한다. 이하에서는, 각 퍼지 추출기의 구체적인 알고리즘을 설명하기로 한다.

[0062] **(1) LWE 기반의 퍼지 추출기**

[0063] 본 기법에서는 엠블럼 인코딩/디코딩 함수 *EC/DC*를 사용한다. 인코딩 함수 *EC*의 입력값 (r,d,q) 중에서 에러의 크기 d와 모듈러스 q는 이미 시스템 파라미터로 주어진 값으로 간주하여, 본 발명에서는 $EC_{d,q}$의 입력값으로 비트열 $r\in\{0,1\}^m$만 입력한다. 디코딩 함수 DC_q도 마찬가지로 입력값 q를 고정된 값으로 간주하여 실제

입력값으로는 벡터 $c \in \mathbb{Z}_q^m$ 만 입력한다.

[0064] **생성(Gen)**

[0065] 생성 단계에서는 l 번의 측정을 통해 생성한 가변 길이 퍼지 데이터들 $w_1 \in \{0,1\}^{n_1}, \cdots, w_l \in \{0,1\}^{n_l}$ 을 입력으로 받아 아래와 같이 사용자의 헬퍼 데이터(Helper data) P 와 랜덤 비트열 $k \in \{0,1\}^{\lambda}$ 를 생성한다. 여기서, λ 는 보안 상수(security parameter)를 의미할 수 있다. 또한, 랜덤 비트열 k 는 비밀키(secret key) 등과 같은 암호학적인 키로 사용 가능하며, 헬퍼 데이터 P 는 복원(Rec) 단계에서 동일한 랜덤 비트열을 생성하기 위해 이용되는 데이터이다.

[0066] 1. 랜덤 비트열 $k \in \{0,1\}^{\lambda}$ 을 선택한다.

[0067] 2. 모든 $i = 1, \cdots, l$ 에 대해 아래 과정을 수행한다.

[0068] (a) 랜덤 행렬 $A_i \leftarrow [-L_i, L_i]^{m \times n_i}$ 를 선택한다. 자연수 $L_i (< q)$ 는 n_i 에 의해 결정된다. 즉, 격자(Lattice)를 나타내는 행렬 A_i 의 원소들은 비교적 작은 값들로 구성될 수 있다. 예시적으로, L_i 은 7이고 q 는 13일 수 있다.

[0069] (b) 이산 정규 분포(가우시안 분포) χ_i^m 에서 랜덤한 벡터 e_i 를 선택한다($e_i \leftarrow \chi_i^m$).

[0070] (c) 랜덤 비트열 $k_i \in \{0,1\}^{\lambda}$ 를 선택한 후, $r_i = \mathbf{ECC.Encode}(k_i)$ 를 계산한다.

[0071] (d) $b_i = A_i w_i + e_i + EC_{d,q}(r_i) \bmod q$ 을 계산한다. 이때, $EC_{d,q}$ 는 엠블럼 인코딩이다.

[0072] (e) 의사 난수 생성기 $\mathbf{PRG}$ 를 이용하여 $(k_{i1}, k_{i2}) = \mathbf{PRG}(k_i)$ 를 생성한다.

[0073] (f) 대칭키 암호 알고리즘 $\mathbf{SYM} = (\mathbf{SYM.Encrypt}, \mathbf{SYM.Decrypt})$ 을 이용하여 k 를 $c_i = \mathbf{SYM.Encrypt}(k_{i1}, k)$ 로 암호화한다.

[0074] (g) 메시지 인증 코드 $\mathbf{MAC}$ 를 이용하여 $m_i = \mathbf{MAC}(k_{i2}, b_i \| c_i)$ 를 생성한다.

[0075] 3. 헬퍼 데이터 $P = (\{A_i, b_i, m_i, c_i\}_{i \in [l]})$ 와 랜덤 비트열 k 를 출력한다.

[0076] **복원(Rec)**

[0077] 복원 단계에서는 퍼지 데이터 $w' \in \{0,1\}^n$ 와 헬퍼 데이터 $P = (\{A_i, b_i, m_i, c_i\}_{i \in [l]})$ 를 입력으로 받아 아래와 같이 비트열 $k' \in \{0,1\}^{\lambda}$ 을 생성(또는 복원)한다.

[0078] 1. 헬퍼 데이터에서 $n_i = n$ 을 만족하는 첫 번째 인덱스 i 를 선택한 후 아래 과정을 수행한다. 만약 존재하지 않는다면 새로운 퍼지 데이터를 입력받아 1번부터 다시 수행한다.

[0079] (a) $r'_i = DC_q(b_i - A_i w' \bmod q)$ 를 계산한다. 이때, DC_q 는 엠블럼 디코딩이다.

[0080] (b) 비트열 $k'_i = \mathbf{ECC.Decode}(r'_i)$ 를 출력한다.

[0081] (c) 의사 난수 생성기 $\mathbf{PRG}$ 를 이용하여 $(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i)$ 를 생성한다.

[0082] (d) 메시지 인증 코드 $\mathbf{MAC}$ 를 이용하여 $m_i' = \mathbf{MAC}(k_{i2}', b_i \| c_i)$ 를 생성한다.

[0083] (e) 생성한 m_i'가 m_i와 같은지 확인한다. 같으면 $k' = \mathbf{SYM.Decrypt}(k'_{i1}, c_i)$을 생성하여 출력한다. 다르면 $n_i = n$를 만족하는 다음 인덱스로 (a)부터 다시 수행한다. 조건을 만족하는 인덱스가 없으면 새로운 퍼지 데이터를 입력받아 1부터 다시 수행한다.

[0084] 복원 단계에서 입력된 퍼지 데이터 $w' \in \{0,1\}^n$가 생성 단계에서 사용한 가변 길이 퍼지 데이터들 $w_1 \in \{0,1\}^{n_1}, \cdots, w_t \in \{0,1\}^{n_t}$ 중 하나인 w_i와 퍼지 데이터의 길이가 같고, w_i와 w'의 데이터 간 거리가 임계치보다 작으면, 복원 단계에서 출력한 비트열 k'가 k와 같음을 아래 과정들을 통해 확인할 수 있다.

[0085] (a)에서 생성한 $r'_i = DC_q(b_i - A_i w' \bmod q)$은 아래 수식을 통해 r_i와 같음을 알 수 있다.

[0086]
$$r'_i = DC_q(b_i - A_i w' \bmod q) = DC_q(A_i w_i + e_i + EC_{d,q}(r_i) - A_i w' \bmod q)$$
$$= DC_q(A_i(w_i - w') + e_i + EC_{d,q}(r_i) \bmod q) = r_i$$

[0087] 이때, $r_i = \mathbf{ECC.Encode}(k_i)$임을 이용하면, (b)에서 생성한 $k'_i = \mathbf{ECC.Decode}(r'_i)$가 k_i와 같음을 아래 수식을 통해 알 수 있다.

[0088] $$k'_i = \mathbf{ECC.Decode}(r'_i) = \mathbf{ECC.Decode}(r_i) = k_i$$

[0089] 이에 따라, (c)에서 생성한 $(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i)$는 (k_{i1}, k_{i2})와 같음을 아래 수식을 통해 알 수 있다.

[0090] $$(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i) = \mathbf{PRG}(k_i) = (k_{i1}, k_{i2})$$

[0091] 따라서 (d)에서 생성한 $m_i' = \mathbf{MAC}(k_{i2}', b_i \| c_i)$는 $m_i = \mathbf{MAC}(k_{i2}, b_i \| c_i)$와 같으므로 (e)에서 $k' = \mathbf{SYM.Decrypt}(k'_{i1}, c_i)$을 계산하게 된다. 이때, $c_i = \mathbf{SYM.Encrypt}(k_{i1}, k)$이고, $k_{i1}' = k_{i1}$이므로 생성한 k'는 아래 수식을 통해 k와 같음을 알 수 있다.

[0092]
$$k' = \mathbf{SYM.Decrypt}(k'_{i1}, c_i) = \mathbf{SYM.Decrypt}(k'_{i1}, \mathbf{SYM.Encrypt}(k_{i1}, k))$$
$$= \mathbf{SYM.Decrypt}(k_{i1}, \mathbf{SYM.Encrypt}(k_{i1}, k)) = k$$

[0094] **(2) LWR 기반의 퍼지 추출기**

[0095] 본 기법에서는 엠블럼 인코딩/디코딩 함수(*EC/DC*)를 이용한다. 인코딩 함수 *EC*의 입력값 (r, d, p) 중에서 에러의 크기 d와 모듈러스 p는 이미 시스템 파라미터로 주어진 값으로 간주하여, 본 발명에서는 $EC_{d,p}$의 입력값으로 비트열 $r \in \{0,1\}^m$만 입력한다. 디코딩 함수 DC_q도 마찬가지로 입력값 q를 고정된 값으로 간주하여 실제 입력값으로는 벡터 $c \in \mathbb{Z}_q^m$만 입력한다. 또한, 본 기법에서는 $q/p = 2^t$ (t는 양의 정수)로 정의한다. LWR 기반의 퍼지 추출기의 구체적인 단계는 다음과 같다.

[0096] <u>생성(Gen)</u>

[0097] 생성 단계에서는 l번(l은 임의의 자연수)의 측정을 통해 생성한 가변 길이 퍼지 데이터들 $w_1 \in \{0,1\}^{n_1}, \cdots, w_l \in \{0,1\}^{n_l}$을 입력으로 받아 아래와 같이 사용자의 헬퍼 데이터 P와 랜덤 비트열 $k \in \{0,1\}^{\lambda}$를 생성한다. 여기서, λ는 보안 상수를 의미할 수 있다. 또한, 랜덤 비트열 k는 비밀키(secret key) 등과 같은 암호학적인 키로 사용가능하며, 헬퍼 데이터 P는 복원(Rec) 단계에서 동일한 랜덤 비트열을 생성하기 위해 이용되는 데이터이다.

[0098] 1. 랜덤 비트열 $k \in \{0,1\}^{\lambda}$을 선택한다.

[0099] 2. 모든 $i = 1,\cdots,l$에 대해 아래 과정을 수행한다.

[0100] (a) 랜덤 비트열 $A_i \leftarrow [-L_i, L_i]^{m \times n_i}$를 선택한다. 자연수 $L_i (< q)$는 n_i에 의해 결정된다.

[0101] (b) 랜덤 비트열 $k_i \in \{0,1\}^{\lambda}$를 선택한 후, $r_i = \mathbf{ECC.Encode}(k_i)$를 계산한다.

[0102] (c) $\lfloor A_i w_i \bmod q \rceil_p + EC_{d,p}(r_i) \bmod p$을 계산한다. 이때, $EC_{d,p}$는 엠블럼 인코딩이다.

[0103] (d) 의사 난수 생성기 $\mathbf{PRG}$를 이용하여 $(k_{i1}, k_{i2}) = \mathbf{PRG}(k_i)$를 생성한다.

[0104] (e) 대칭키 암호 알고리즘 $\mathbf{SYM} = (\mathbf{SYM.Encrypt}, \mathbf{SYM.Decrypt})$을 이용하여 k를 $c_i = \mathbf{SYM.Encrypt}(k_{i1}, k)$로 암호화한다.

[0105] (f) 메시지 인증 코드 $\mathbf{MAC}$를 이용하여 $m_i = \mathbf{MAC}(k_{i2}, b_i \| c_i)$를 생성한다.

[0106] 2. 헬퍼 데이터 $P = (\{A_i,\ b_i,\ m_i,\ c_i\}_{i \in [l]})$와 랜덤 비트열 k를 출력한다.

[0107] **복원(Rec)**

[0108] 복원 단계에서는 퍼지 데이터 $w' \in \{0,1\}^n$와 헬퍼 데이터 $P = (\{A_i,\ b_i,\ m_i,\ c_i\}_{i \in [l]})$를 입력으로 받아 다음과 같이 비트열 $k' \in \{0,1\}^{\lambda}$을 생성(또는 복원)한다.

[0109] 1. 헬퍼 데이터에서 $n_i = n$를 만족하는 첫 번째 인덱스 i를 선택한다. 만약 존재하지 않는다면 새로운 퍼지 데이터를 입력받아 1번부터 다시 수행한다.

[0110] (a) $r'_i = DC_q\left(\left(\frac{q}{p}\right) b_i - A_i w' \bmod q\right)$를 계산한다. 여기서, DC_q는 엠블럼 디코딩이다.

[0111] (b) 비트열 $k'_i = \mathbf{ECC.Decode}(r'_i)$를 출력한다.

[0112] (c) 의사 난수 생성기 $\mathbf{PRG}$를 이용하여 $(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i)$를 생성한다.

[0113] (d) 메시지 인증 코드 $\mathbf{MAC}$를 이용하여 $m_i' = \mathbf{MAC}(k_{i2}', b_i \| c_i)$를 생성한다.

[0114] (e) 생성한 m_i'가 m_i와 같은지 확인한다. 확인 결과, 두 값이 같으면 $k' = \mathbf{SYM.Decrypt}(k'_{i1}, c_i)$을 생성하여 출력한다. 다르면 $n_i = n$를 만족하는 다음 인덱스로 (a)부터 다시 수행한다. 조건을 만족하는 인덱스가 없으면 새로운 퍼지 데이터를 입력받아 1부터 다시 수행한다.

[0115] 복원 단계에서 입력된 퍼지 데이터 $w' \in \{0,1\}^n$가 생성 단계에서 사용한 가변 길이 퍼지 데이터들 $w_1 \in \{0,1\}^{n_1}, \cdots, w_t \in \{0,1\}^{n_t}$ 중 하나인 w_i와 퍼지 데이터의 길이가 같고, w_i와 w'의 데이터 간 거리가 임계치보다 작다면, 복원 단계에서 출력한 비트열 k'가 k와 같음을 아래 과정들을 통해 확인할 수 있다.

[0116] (a)에서 생성한 $r'_i = DC_q\left(\left(\frac{q}{p}\right)b_i - A_i w' \bmod q\right)$ 은 아래 수식을 통해 r_i와 같음을 알 수 있다.

$$
\begin{aligned}
r_i' &= DC_q\left(\left(\frac{q}{p}\right)b_i - A_i w' \bmod q\right) \\
&= DC_q\left(\left(\frac{q}{p}\right)\lfloor A_i w_i \bmod q \rfloor_p + \left(\frac{q}{p}\right)EC_{d,p}(r_i) - A_i w' \bmod q\right) \\
&= DC_q\left(\left(\frac{q}{p}\right)\lfloor A_i w_i \bmod q \rfloor_p - A_i w' + \left(\frac{q}{p}\right)EC_{d,p}(r_i) \bmod q\right)
\end{aligned}
$$

[0117] $= r$

[0118] 이때, $r_i = \mathbf{ECC.Encode}(k_i)$임을 이용하면, (b)에서 생성한 $k'_i = \mathbf{ECC.Decode}(r'_i)$가 k_i와 같음을 아래 수식을 통해 알 수 있다.

[0119] $$k'_i = \mathbf{ECC.Decode}(r'_i) = \mathbf{ECC.Decode}(r_i) = k_i$$

[0120] 이에 따라, (c)에서 생성한 $(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i)$는 (k_{i1}, k_{i2})와 같음을 아래 수식을 통해 알 수 있다.

[0121] $$(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i) = \mathbf{PRG}(k_i) = (k_{i1}, k_{i2})$$

[0122] 따라서 (d)에서 생성한 $m_i' = \mathbf{MAC}(k_{i2}', b_i \| c_i)$는 $m_i = \mathbf{MAC}(k_{i2}, b_i \| c_i)$와 같으므로 (e)에서 $k' = \mathbf{SYM.Decrypt}(k'_{i1}, c_i)$을 계산하게 된다. 이때, $c_i = \mathbf{SYM.Encrypt}(k_{i1}, k)$이고, $k_{i1}' = k_{i1}$이므로 생성한 k'는 아래 수식을 통해 k와 같음을 알 수 있다.

[0123] $$
\begin{aligned}
k' &= \mathbf{SYM.Decrypt}(k'_{i1}, c_i) = \mathbf{SYM.Decrypt}(k'_{i1}, \mathbf{SYM.Encrypt}(k_{i1}, k)) \\
&= \mathbf{SYM.Decrypt}(k_{i1}, \mathbf{SYM.Encrypt}(k_{i1}, k)) = k
\end{aligned}
$$

[0125] 도 3은 본 발명의 일 실시예에 따른 퍼지 추출 시스템을 도시한다.

[0126] 도 3을 참조하면, 인증 시스템, 키 추출 시스템, 키 복구 시스템 등으로 명명될 수도 있는 퍼지 추출 시스템(10)은 제1 단말(100), 제2 단말(300), 및 서버(500)를 포함한다.

[0127] 제1 단말(100)은 사용자의 제1 퍼지 데이터들을 이용하여 제1 키와 상기 제1 키를 복구하는 과정에서 이용되는 헬퍼 데이터를 생성한다. 생성된 헬퍼 데이터는 서버(500)로 송신될 수 있으며, 상기 제1 키는 제1 단말(100) 내에서 삭제될 수 있다. 또한, 상기 헬퍼 데이터는 상기 사용자를 식별할 수 있는 식별 정보(예컨대, 상기 사용자의 ID)와 함께 상기 서버(500)로 송신될 수도 있다. 실시예에 따라, 제1 단말(100)과 서버(500) 사이에서 상기 사용자를 인증하기 위한 과정이 추가될 수도 있다.

[0128] 서버(500)는 제1 단말(100)로부터 수신된 상기 헬퍼 데이터를 상기 사용자의 식별 정보와 함께 서버(500) 내에 구비된 저장 장치 또는 서버(500)와는 별도로 분리되어 구현된 저장 장치에 저장할 수 있다. 서버(500)는 상기 헬퍼 데이터를 요청하는 임의의 단말, 예컨대 제2 단말(500)의 요청에 응답하여 상기 헬퍼 데이터를 상기 임의의 단말로 송신할 수 있다. 상기 임의의 단말은 상기 헬퍼 데이터를 이용하여 상기 제1 단말이 생성한 상기 제1 키와 동일한 제2 키를 생성(또는 복구)할 수 있다.

[0129] 제2 단말(300)은 사용자의 제2 퍼지 데이터와 서버(500)로부터 수신된 상기 헬퍼 데이터를 이용하여 상기 제1 키와 동일한 제2 키를 생성(또는 복구)할 수 있다. 구체적으로, 제2 단말(300)은 상기 사용자의 식별 정보(예컨대, 상기 사용자의 ID)를 포함하는 헬퍼 데이터 요청 메시지를 서버(500)로 송신하고, 서버(500)는 상기 헬퍼 데이터 요청 메시지에 응답하여 상기 사용자의 식별 정보에 대응하는 헬퍼 데이터를 제2 단말(300)로 송신할 수 있다. 실시예에 따라, 서버(500)는 상기 사용자에 대한 인증 과정을 수행한 후에, 인증에 성공한 경우에만 상기 헬퍼 데이터를 상기 제2 단말(300)로 송신할 수 있다. 또한, 실시예에 따라 제2 단말(300)은 제1 단말(100)을

의미할 수도 있다.

[0130] 제1 단말(100), 제2 단말(300), 및 서버(500) 각각은 각종 연산 처리 및 신호 생성이 가능한 적어도 하나의 전자 장치를 포함할 수 있다. 여기서, 적어도 하나의 전자 장치는 프로세서 및/또는 프로세서가 설치된 컴퓨팅 장치를 포함할 수 있다. 프로세서는 중앙 처리 장치(CPU, Central Processing Unit), 마이크로 컨트롤러 유닛(MCU, Micro Controller Unit), 마이컴(Micom, Micro Processor), 애플리케이션 프로세서(AP, Application Processor), 전자 제어 유닛(ECU, Electronic Controlling Unit), 그래픽 처리 장치(GPU, Graphic Processing Unit) 및/또는 각종 연산 처리 및 제어 신호의 생성이 가능한 처리 장치 등을 포함할 수 있다. 이들 처리 장치는 하나 또는 둘 이상의 반도체 칩 및 관련 부품을 이용하여 구현될 수 있다. 또한, 컴퓨팅 장치는 데스크톱 컴퓨터, 랩톱 컴퓨터, 서버용 컴퓨터, 스마트 폰, 태블릿 피씨, 스마트 시계, 두부 장착형 디스플레이(HMD, Head Mounted Display) 장치, 휴대용 게임기, 내비게이션 장치, 개인용 디지털 보조기(PDA, Personal Digital Assistant), 인공지능 스피커 장치, 디지털 텔레비전, 셋톱 박스, 로봇, 가전 기기, ATM(Automated Teller Machine), 벤딩 머신(Vending machine), 기계 장치 및/또는 이외 정보 처리 기능을 수행할 수 있는 적어도 하나의 전자 장치를 포함할 수 있다.

[0131] 제1 단말(100), 제2 단말(300), 및 서버(500) 각각은 저장부와 데이터 송수신이 가능하도록 직접 또는 간접적으로 연결되어 있을 수 있다. 저장부는 제1 단말(100)에 의한 헬퍼 데이터 생성 과정에서 이용되거나 이를 이용하여 생성되는 데이터, 제2 단말(300)에 의한 제2 키 복구 과정에서 이용되거나 이를 이용하여 생성되는 데이터, 또는 서버(500)에 저장되는 각 사용자의 식별 정보와 헬퍼 데이터를 일시적으로 또는 비일시적으로 저장할 수 있다. 저장부는 주기억장치 및 보조기억장치 중 적어도 하나를 포함할 수 있으며, 주기억장치는 롬(ROM) 및/또는 램(RAM) 등을 포함할 수 있고, 보조기억장치는 플래시 메모리 장치, SD(Secure Digital) 카드, 솔리드 스테이트 드라이브(SSD, Solid State Drive), 하드 디스크 드라이브(HDD, Hard Disc Drive) 컴팩트 디스크(CD), 디브이디(DVD) 및/또는 레이저 디스크 등을 포함할 수 있다.

[0132] 제1 단말(100), 제2 단말(300), 및 서버(500) 각각은 저장부에 저장된 애플리케이션을 구동시켜, 헬퍼 데이터의 생성, 제2 키의 복구 등 다양한 동작을 수행할 수도 있다. 여기서, 애플리케이션은, 설계자에 의해 직접 작성되어 저장부에 저장 또는 갱신된 것일 수도 있고, 또는 유선 또는 무선 통신 네트워크를 통해 접속 가능한 전자 소프트웨어 유통망 등을 통하여 획득 또는 갱신된 것일 수도 있다.

[0134] 도 4는 도 3에 도시된 제1 단말의 기능 블럭도이다.

[0135] 도 4를 참조하면, 제1 단말(100)은 퍼지 데이터 수신부(110), 키 생성부(120), 헬퍼 데이터 생성부(130), 헬퍼 데이터 송신부(140), 및 저장부(150) 중 적어도 하나 이상을 포함할 수 있다. 제1 단말(100)은 사용자의 퍼지 데이터들을 이용하여 제1 키와 상기 제1 키를 복구하는 과정에서 필요한 헬퍼 데이터를 생성하고, 상기 헬퍼 데이터를 서버(500)로 송신할 수 있다.

[0136] 퍼지 데이터 수신부(110)는 소정의 입출력 장치를 이용하여 사용자로부터 지문 정보, 홍채 정보 등과 같은 생체 정보를 수신하고, 임의의 변환 알고리즘을 이용하여 수신된 생체 정보를 비트열 형태로 변환할 수 있다. 사용자의 생체 정보의 수신과 비트열 형태의 변환 동작을 여러 차례 수행될 수 있다. 이 경우, 비트열 형태의 퍼지 데이터들 각각은 서로 동일한 길이를 갖거나 서로 상이한 길이를 가질 수 있다. 즉, 각 퍼지 데이터의 길이는 미리 정해진 길이를 갖지 않으며 생체 정보를 수신할 때마다 또는 비트열 형태로 변환할 때마다 상이한 길이를 갖는 퍼지 데이터가 생성될 수 있다. 또한, 실시예에 따라 퍼지 데이터 수신부(110)는 비트열 형태로 변환된 퍼지 데이터를 복수회 수신할 수도 있다. 이 경우에도, 수신된 퍼지 데이터들 각각의 길이는 동일할 수도 있고 상이할 수도 있다. 즉, 본 발명에서 말하는 퍼지 데이터는 가변 길이의 비트열 형태로 변환된 데이터를 의미할 수 있다.

[0137] 퍼지 데이터 수신부(110)에 의해 수신된 퍼지 데이터들 및/또는 비트열로 변환된 퍼지 데이터들은 저장부(150)에 저장될 수 있다. 실시예에 따라, 사용자의 퍼지 데이터들은 미리 저장부(150)에 저장될 수도 있고, 이 경우에 퍼지 데이터 수신부(110)는 제1 단말(100)에서 생략될 수도 있다.

[0138] 키 생성부(120)는 랜덤 비트열($k \in \{0,1\}^{\lambda}$)을 선택할 수 있고, 상기 랜덤 비트열은 제1 키를 의미할 수 있다. 키 생성부(120)에 의해 생성된 랜덤 비트열은 저장부(150)에 저장될 수 있다.

[0139] 헬퍼 데이터 생성부(130)는 앞서 설명한 2가지의 기법 중 적어도 하나를 이용하여 헬퍼 데이터 P를 생성할 수

있다. 생성된 헬퍼 데이터는 저장부(150)에 저장될 수 있다.

[0140] (1) LWE 기반의 퍼지 추출 기법을 이용하는 경우, 헬퍼 데이터 생성부(130)는 모든 $i=1,\cdots,l$에 대해 아래 과정을 수행함으로써 헬퍼 데이터를 생성할 수 있다.

[0141] (a) 랜덤 비트열 $A_i \leftarrow [-L_i, L_i]^{m \times n_i}$를 선택한다. 자연수 $L_i (< q)$는 n_i에 의해 결정된다.

[0142] (b) 랜덤 비트열 $k_i \in \{0,1\}^{\lambda}$를 선택한 후, $r_i = \mathbf{ECC.Encode}(k_i)$를 계산한다.

[0143] (c) $\lfloor A_i w_i \bmod q \rceil_p + EC_{d,p}(r_i) \bmod p$을 계산한다. 이때, $EC_{d,p}$는 엠블럼 인코딩이다.

[0144] (d) 의사 난수 생성기 $\mathbf{PRG}$를 이용하여 $(k_{i1}, k_{i2}) = \mathbf{PRG}(k_i)$를 생성한다.

[0145] (e) 대칭키 암호 알고리즘 $\mathbf{SYM} = (\mathbf{SYM.Encrypt}, \mathbf{SYM.Decrypt})$을 이용하여 k를 $c_i = \mathbf{SYM.Encrypt}(k_{i1}, k)$로 암호화한다.

[0146] (f) 메시지 인증 코드 $\mathbf{MAC}$를 이용하여 $m_i = \mathbf{MAC}(k_{i2}, b_i \| c_i)$를 생성한다.

[0147] 즉, 헬퍼 데이터 생성부(130)는 헬퍼 데이터 $P = (\{A_i,\ b_i,\ m_i,\ c_i\}_{i \in [l]})$를 생성하고, 생성된 헬퍼 데이터는 저장부(150)에 저장될 수 있다.

[0148] (2) LWR 기반의 퍼지 추출 기법을 이용하는 경우, 헬퍼 데이터 생성부(130)는 모든 $i=1,\cdots,l$에 대해 아래 과정을 수행함으로써 헬퍼 데이터를 생성할 수 있다.

[0149] (a) 랜덤 비트열 $A_i \leftarrow [-L_i, L_i]^{m \times n_i}$를 선택한다. 자연수 $L_i (< q)$는 n_i에 의해 결정된다.

[0150] (b) 랜덤 비트열 $k_i \in \{0,1\}^{\lambda}$를 선택한 후, $r_i = \mathbf{ECC.Encode}(k_i)$를 계산한다.

[0151] (c) $\lfloor A_i w_i \bmod q \rceil_p + EC_{d,p}(r_i) \bmod p$을 계산한다. 이때, $EC_{d,p}$는 엠블럼 인코딩이다.

[0152] (d) 의사 난수 생성기 $\mathbf{PRG}$를 이용하여 $(k_{i1}, k_{i2}) = \mathbf{PRG}(k_i)$를 생성한다.

[0153] (e) 대칭키 암호 알고리즘 $\mathbf{SYM} = (\mathbf{SYM.Encrypt}, \mathbf{SYM.Decrypt})$을 이용하여 k를 $c_i = \mathbf{SYM.Encrypt}(k_{i1}, k)$로 암호화한다.

[0154] (f) 메시지 인증 코드 $\mathbf{MAC}$를 이용하여 $m_i = \mathbf{MAC}(k_{i2}, b_i \| c_i)$를 생성한다.

[0155] 즉, 헬퍼 데이터 생성부(130)는 헬퍼 데이터 $P = (\{A_i,\ b_i,\ m_i,\ c_i\}_{i \in [l]})$를 생성하고, 생성된 헬퍼 데이터는 저장부(150)에 저장될 수 있다.

[0156] 헬퍼 데이터 송신부(140)는 생성된 헬퍼 데이터를 서버(500)로 송신할 수 있다. 이때, 헬퍼 데이터 송신부(140)는 사용자의 식별 정보를 서버(500)로 송신할 수도 있다.

[0157] 저장부(150)에는 퍼지 데이터 수신부(110)에 의해 수신된 퍼지 데이터, 고정된 길의의 비트열로 변환된 퍼지 데이터, 키 생성부(120)에 의해 선택된 랜덤 비트열, 오류 정정 코드의 인코딩 결과, 헬퍼 데이터 생성부(130)에 의해 생성된 헬퍼 데이터, 및 헬퍼 데이터를 생성하는 과정 중에 생성되는 데이터 등이 일시적으로 또는 비일시적으로 저장될 수 있다. 다만, 사용자의 생체 데이터 및/또는 랜덤 비트열은 사용자의 프라이버시 보호를 위해 제1 단말(100), 즉 저장부(150) 내에서 삭제되는 것이 바람직하다.

[0159] 도 5는 도 3에 도시된 제2 단말의 기능 블럭도이다.

[0160] 도 5를 참조하면, 제2 단말(300)은 퍼지 데이터 수신부(310), 헬퍼 데이터 수신부(320), 키 복구부(330), 및 저장부(340) 중 적어도 하나 이상을 포함할 수도 있다.

[0161] 퍼지 데이터 수신부(310)는 소정의 입출력 장치를 이용하여 사용자로부터 지문 정보, 홍채 정보 등과 같은 생체 정보를 수신하고, 임의의 변환 알고리즘을 이용하여 수신된 생체 정보를 비트열 형태로 변환할 수 있다. 실시예에 따라, 퍼지 데이터 수신부(310)는 비트열 형태로 변환된 퍼지 데이터를 수신할 수도 있다. 즉, 본 발명에서 말하는 퍼지 데이터는 비트열 형태로 변환된 데이터를 의미할 수 있다. 퍼지 데이터 수신부(310)에 의해 수신된 퍼지 데이터 및/또는 비트열로 변환된 퍼지 데이터는 저장부(340)에 저장될 수 있다. 실시예에 따라, 퍼지 데이터 수신부(310)는 사용자의 식별 정보를 수신할 수도 있다.

[0162] 헬퍼 데이터 수신부(320)는 서버(500)로부터 사용자의 헬퍼 데이터를 수신한다. 이를 위해, 헬퍼 데이터 수신부(320)는 사용자의 식별 정보를 포함하는 헬퍼 데이터 요청을 서버(500)로 송신할 수 있다. 실시예에 따라, 소정의 사용자 인증 동작이 수행될 수도 있다. 서버(500)는 사용자의 식별 정보에 대응하는 헬퍼 데이터를 헬퍼 데이터 수신부(320)로 송신할 수 있다. 헬퍼 데이터 수신부(320)에 의해 수신된 헬프 데이터는 저장부(340)에 저장될 수 있다.

[0163] 키 복구부(330)는 앞서 설명한 2가지 기법 중에서 헬퍼 데이터를 생성하는 과정에 사용된 기법을 이용하여 사용자의 퍼지 데이터와 헬퍼 데이터로부터 제2 키를 복구할 수 있다.

[0164] (1) LWE 기반의 퍼지 추출 기법을 이용하는 경우, 키 복구부(330)는 헬퍼 데이터에서 $n_i = n$을 만족하는 첫 번째 인덱스 i를 선택한 후 아래 과정을 수행한다. 만약 존재하지 않는다면, 퍼지 데이터 수신부(310)는 새로운 퍼지 데이터를 수신하고 키 복구부(330)는 새로운 퍼지 데이터를 이용하여 키 복구 과정을 수행한다.

[0165] (a) $r'_i = DC_q(b_i - A_i w' \bmod q)$를 계산한다. 이때, DC_q는 엠블럼 디코딩이다.

[0166] (b) 비트열 $k'_i = \mathbf{ECC.Decode}(r'_i)$를 출력한다.

[0167] (c) 의사 난수 생성기 $\mathbf{PRG}$를 이용하여 $(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i)$를 생성한다.

[0168] (d) 메시지 인증 코드 $\mathbf{MAC}$를 이용하여 $m_i' = \mathbf{MAC}(k_{i2}', b_i \| c_i)$를 생성한다.

[0169] (e) 생성한 m_i'가 m_i와 같은지 확인한다. 같으면 $k' = \mathbf{SYM.Decrypt}(k'_{i1}, c_i)$을 생성하여 출력한다. 다르면 $n_i = n$를 만족하는 다음 인덱스로 (a)부터 다시 수행한다. 조건을 만족하는 인덱스가 없는 경우, 퍼지 데이터 수신부(310)는 새로운 퍼지 데이터를 수신하고, 키 복구부(330)는 새로운 퍼지 데이터를 이용하여 키 복구 과정을 수행한다.

[0170] 복원 단계에서 입력된 퍼지 데이터 $w' \in \{0,1\}^n$가 생성 단계에서 사용한 가변 길이 퍼지 데이터들 $w_1 \in \{0,1\}^{n_1}, \cdots, w_t \in \{0,1\}^{n_t}$ 중 하나인 w_i와 퍼지 데이터의 길이가 같고, w_i와 w'의 데이터 간 거리가 임계치보다 작으면, 복원 단계에서 출력한 비트열 k'가 k와 같다.

[0171] (3) LWR 기반의 퍼지 추출 기법을 이용하는 경우, 키 복구부(330)는 헬퍼 데이터에서 $n_i = n$를 만족하는 첫 번째 인덱스 i를 선택한다. 만약 존재하지 않는다면, 퍼지 데이터 수신부(310)는 새로운 퍼지 데이터를 수신하고, 키 복구부(330)는 새로운 퍼지 데이터를 이용하여 키 복구 과정을 수행한다.

[0172] (a) $r'_i = DC_q\left(\left(\frac{q}{p}\right) b_i - A_i w' \bmod q\right)$를 계산한다. 여기서, DC_q는 엠블럼 디코딩이다.

[0173] (b) 비트열 $k'_i = \mathbf{ECC.Decode}(r'_i)$를 출력한다.

[0174] (c) 의사 난수 생성기 $\mathbf{PRG}$를 이용하여 $(k_{i1}', k_{i2}') = \mathbf{PRG}(k'_i)$를 생성한다.

[0175] (d) 메시지 인증 코드 $\mathbf{MAC}$를 이용하여 $m_i' = \mathbf{MAC}(k_{i2}', b_i \| c_i)$를 생성한다.

[0176] (e) 생성한 m_i'가 m_i와 같은지 확인한다. 확인 결과, 두 값이 같으면 $k' = \mathbf{SYM.Decrypt}(k'_{i1}, c_i)$을 생성하여 출력한다. 다르면 $n_i = n$를 만족하는 다음 인덱스로 (a)부터 다시 수행한다. 조건을 만족하는 인덱스가 없는 경우, 퍼지 데이터 수신부(310)는 새로운 퍼지 데이터를 수신하고, 키 복구부(330)는 새로운 퍼지 데이터를 이용하여 키 복구 과정을 수행한다.

[0177] 복원 단계에서 입력된 퍼지 데이터 $w' \in \{0,1\}^n$가 생성 단계에서 사용한 가변 길이 퍼지 데이터들 $w_1 \in \{0,1\}^{n_1}, \cdots, w_t \in \{0,1\}^{n_t}$ 중 하나인 w_i와 퍼지 데이터의 길이가 같고, w_i와 w'의 데이터 간 거리가 임계치보다 작다면, 복원 단계에서 출력한 비트열 k'가 k와 같다.

[0178] 저장부(340)에는 퍼지 데이터 수신부(310)에 의해 수신된 퍼지 데이터, 비트열로 변환된 퍼지 데이터, 헬퍼 데이터 수신부(320)에 의해 수신된 헬퍼 데이터, 키 복구부(330)에 의해 복구된 제2 키, 상기 제2 키를 복구하는 과정에서 생성되는 데이터가 일시적으로 또는 비일시적으로 저장될 수 있다. 다만, 사용자의 생체 데이터 및/또는 상기 제2 키는 사용자의 프라이버시 보호를 위해 제2 단말(300), 즉 저장부(340) 내에서 삭제되는 것이 바람직하다.

[0180] 이상에서 설명된 장치는 하드웨어 구성 요소, 소프트웨어 구성 요소, 및/또는 하드웨어 구성 요소 및 소프트웨어 구성 요소의 집합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성 요소는, 예를 들어, 프로세서, 콘트롤러, ALU(Arithmetic Logic Unit), 디지털 신호 프로세서(Digital Signal Processor), 마이크로컴퓨터, FPA(Field Programmable array), PLU(Programmable Logic Unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(Operation System, OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술 분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(Processing Element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(Parallel Processor)와 같은, 다른 처리 구성(Processing Configuration)도 가능하다.

[0181] 소프트웨어는 컴퓨터 프로그램(Computer Program), 코드(Code), 명령(Instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(Collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성 요소(Component), 물리적 장치, 가상 장치(Virtual Equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(Signal Wave)에 영구적으로, 또는 일시적으로 구체화(Embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0182] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM, DVD와 같은 광기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-optical Media), 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0183] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성 요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

부호의 설명

[0184] 100 : 제1 단말 110 : 퍼지 데이터 수신부

120 : 키 생성부 130 : 헬퍼 데이터 생성부

140 : 헬퍼 데이터 송신부 150 : 저장부

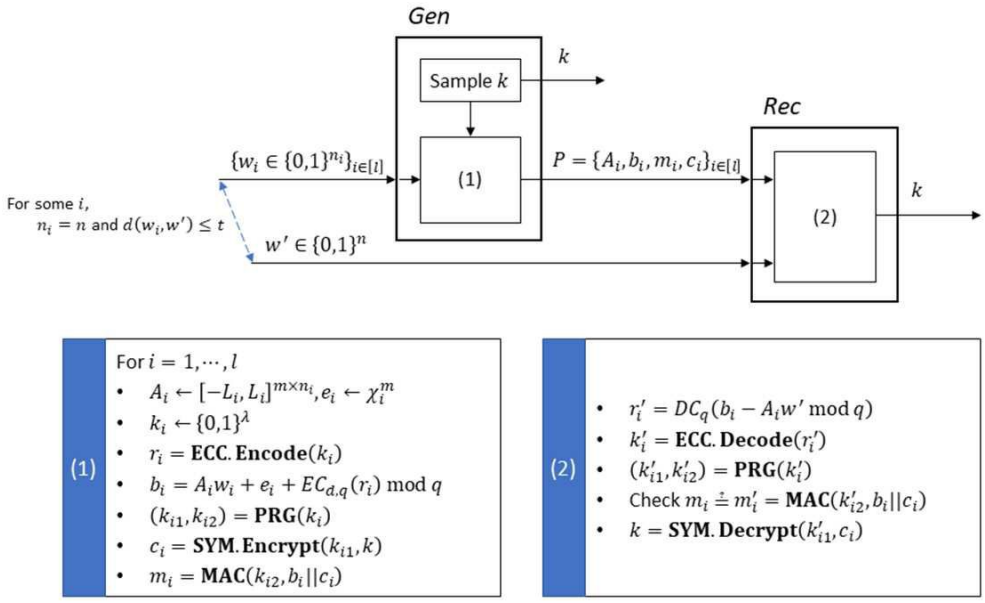
300 : 제2 단말 310 : 퍼지 데이터 수신부

320 : 헬퍼 데이터 수신부 330 : 키 복구부

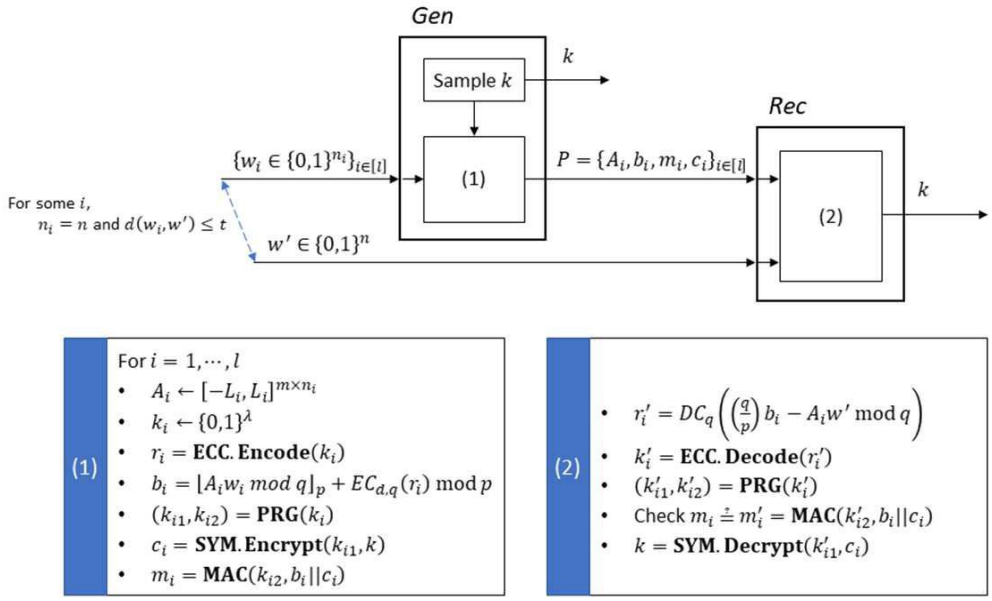
340 : 저장부 500 : 서버

도면

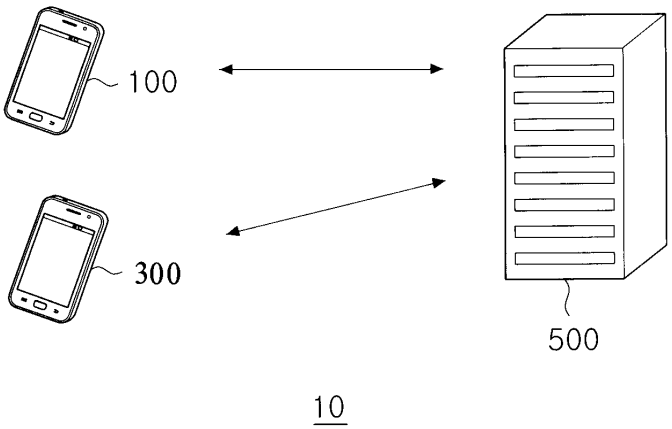
도면1



도면2

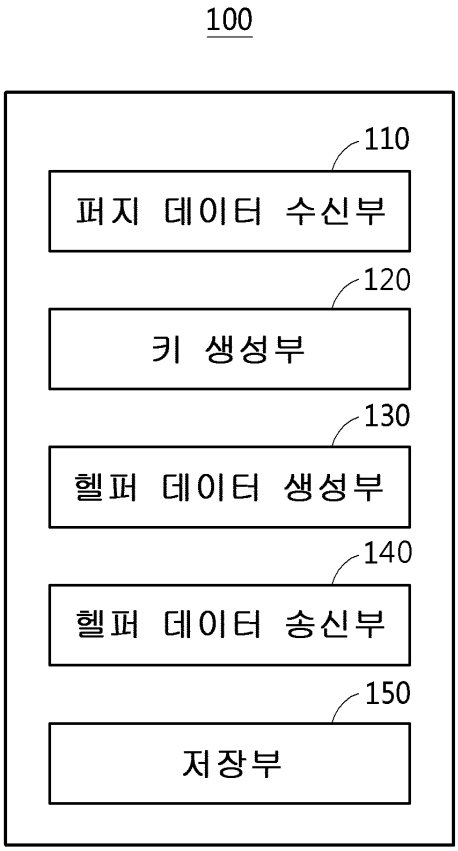

Gen
Sample k
k
Rec
$\{w_i \in \{0,1\}^{n_i}\}_{i\in[l]}$
(1)
$P = \{A_i, b_i, m_i, c_i\}_{i\in[l]}$
(2)
k
For some i,
$n_i = n$ and $d(w_i, w') \leq t$
$w' \in \{0,1\}^n$
(1)
For $i = 1, \cdots, l$
$A_i \leftarrow [-L_i, L_i]^{m\times n_i}$
$k_i \leftarrow \{0,1\}^{\lambda}$
$r_i = \mathbf{ECC.Encode}(k_i)$
$b_i = \lfloor A_i w_i \bmod q \rceil_p + EC_{d,q}(r_i) \bmod p$
$(k_{i1}, k_{i2}) = \mathbf{PRG}(k_i)$
$c_i = \mathbf{SYM.Encrypt}(k_{i1}, k)$
$m_i = \mathbf{MAC}(k_{i2}, b_i || c_i)$
(2)
$r_i' = DC_q\left(\left(\frac{q}{p}\right) b_i - A_i w' \bmod q\right)$
$k_i' = \mathbf{ECC.Decode}(r_i')$
$(k_{i1}', k_{i2}') = \mathbf{PRG}(k_i')$
Check $m_i \stackrel{?}{=} m_i' = \mathbf{MAC}(k_{i2}', b_i || c_i)$
$k = \mathbf{SYM.Decrypt}(k_{i1}', c_i)$

도면3

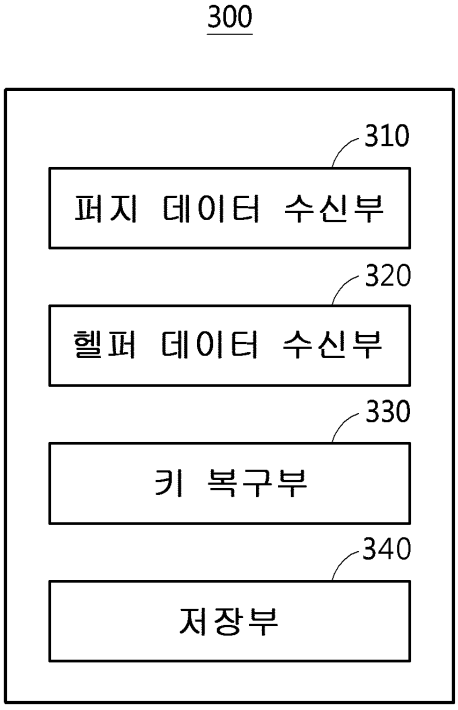

100
300
500

10

도면4


100
110
퍼지 데이터 수신부
120
키 생성부
130
헬퍼 데이터 생성부
140
헬퍼 데이터 송신부
150
저장부

도면5


300
310
퍼지 데이터 수신부
320
헬퍼 데이터 수신부
330
키 복구부
340
저장부